Eur päisches Patentamt | European Patent Office | Office européen des brevets

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | Patent application No. | Demande de brevet n°

02368117.4

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 02368117.4
Demande no:

Anmeldetag:
Date of filing: 25.10.02
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

International Business Corporation

Armonk, NY 10504
ETATS-UNIS D'AMERIQUE

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

A secure system and method for media content data file network distribution

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

H04L29/00

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR

A SECURE SYSTEM AND METHOD FOR MEDIA CONTENT DATA FILE NETWORK DISTRIBUTION

Field of the Invention

The present invention generally relates to network file distribution; more particularly, the present invention concerns security in media file distribution as the name or the network address of the file must not be read by the receiver.

Background of the Invention

Media content servers can be accessed through networks by customer computers to download and play media content files containing digitized music or video sequences or radio broadcast etc.. The customer computer has a software installed, a media player, allowing playback of the media content. Media content is read either in real time, for streaming media content, or from a CDROM or hard disk from media files stored on these storage units. The streaming media players allow decompression of media content and playback in real time towards the louder and the display of the subscriber computer. The software can be integrated in the subscriber computer browser or can be downloaded from the media content servers as a plug-in. The company RealNetworks, Inc. provides such a streaming media player, RealPlayer (a trademark of RealNetworks, Inc. in certain countries). The company Microsoft Corporation provides Windows Media Player program performing the same functions. The media content servers offer also sets of programs to create the media content file and to broadcast them to other service subscribers.

The traffic generated by this industry on the network includes messages exchanged between the servers and the customer computers comprising downloading of data. This

traffic is subject to piracy and the delivery of media content must be secured against access by others than the media content server subscriber. Reducing piracy protects both the service provider's ongoing business and the copyright holder of the content.

To access a media content file such a video file from the browser media player of a customer computer, a HTML page is built which embedded a media content client in the page. The embedding process needs to explicitly refer to the file path in the HTML code. For security reason it is necessary to hide the file path.

Here is an example of embedding a player in a HTML page, the code refers explicitly to the file name and path of a video file for a movie:

Rtsp://m1.media.tele.net:554/archive/film1.rm.

Anyone can view this movie simply copying this link in another player. Moreover, someone can catch the naming convention of the films and try to view other movies of the archive (eg film2, film3...).

An easy mode to catch the naming convention of the films is, for example, to view the source code or to use the video server characteristic which consists in displaying for the client, if a connection is broken or the file is not available, an error dialog box with the full path of the movie file.

One solution to solve this problem is to use '.js' files containing JavaScript functions that builds dynamically the full explicit path and name of the movie files. As Java script files are visible, the name also, if is not included in the

HTML page, can be calculated as well as its location, by reading the ';js' file.

This solution has following disadvantages:

- The '.js' files can be easily downloaded from the Internet site. If a user writes the '.js' URL in the address box of his browser, the browser automatically downloads the file on the user machine,
- If an unregistered user reads the '.js' code, he can easily deduct the name of a media content file,
- If an unregistered user reads the '.js' code, he can catch the name convention (typically static) used in the media content archive,
- If an unregistered user reads the '.js' code can distribute himself this information, cracking the system security.

Finally, the usual typical solution to avoid piracy of the media content file is to encrypt the content of the file. The encryption is performed while storing the file on the media content server data base and the data is decrypted in the customer computer before paying it. The huge problem with this solution is the computer resources required to perform encryption/decryption on files which are usually big files. Also, for improving the security, the encryption key is large and the CPU utilization is high. The servers may have the CPU resources necessary for to encryption but it is not the case of the customer computers. From a business point of view, this strategy is against the objective of the media content providers which is to distribute data to the most of the domestic computers installed. By the way, the encryption solution does not prevent piracy of source where the file resides.

Summary of the Invention

It is therefore an object of the present invention to prevent free access to the media content files to pirate able to collect the messages exchanged on the network between the customer computer and the media content servers.

It is one other object of the invention to have a solution which does not consume too much computer resources of the domestic computers to which the media content data are distributed.

The objects are also reached by the method, according claim 1, for providing a subscriber computer with an access, through a network, to media content data from an application server, said method comprising the steps of:

- opening a session with a subscriber;
- receiving from the subscriber computer a request with a list of at least one media content data name;
- creating a temporary metafile having a temporary name;
- writing in the metafile a file path cyphered according to a cyphering algorithm for each media content data name and writing a corresponding network address where said media content data is stored;
- sending to the subscriber computer the temporary metafile name;
- canceling the metafile at the end of session with the subscriber computer. The metafile is received and read on the subscriber computer which requests the content to the media content server having the network address included in the metafile. The media content server, if it recognizes the cyphered file path, sends back to the subscriber computer the file content to be played.

The objects are also reached by the programs installed on the application server, the subscriber computer and the media content servers containing instructions implementing the method of the invention as claimed in the method claims.

The objects are also reached by the computing system for carrying out the method claims.

The clients and media servers transmit information through an other server that has the scope to provide messages containing the requested videos in a secure way and protected from piracy. A temporary metafile is used in which there are the list of the media. The name of the metafile is dynamic and is dependent from the current session. When the session ends, the temporary metafile is deleted. The advantage of the method, according to the invention, is that the real path of videos and the real naming convention are unknown from the message content.

The solution of the present invention prevents a cracker to learn the media content (such as video) file name and the file name convention. A cracker cannot download the metafile according to the present because, if he writes the URL of the metafile in the address box of his browser, the browser, automatically, will launch the video player plug-in. If the cracker has not the video player plug-in installed, he can download the metafile locally but he cannot decipher the video file name stored in it and learn the file naming convention because the video file name is ciphered.

The only risk is to have the cracker playing the movie during the session of the authorized user. But a metafile name caught by a cracker at a time must be quickly reused for playing as it has a limited life duration. After the session of the authorized user, the metafile is no more available. A same name has no chance to be assigned again as the metafile name itself depends on the session name.

The computer resource consumption, particularly CPU resource consumption, at the customer computer level is quite

reasonable considering the average resources of domestic computers. The solution of the present invention avoids to design and implement complex security scenarios based on protocols. The solution is easy to implement and can be used in all preexisting web applications in which there are repositories of video, vowel, picture files or mix of these files. The method of the present invention based on a temporary bridge file to protect a resource is applicable to a lot of situations such as the transmission on mobile phones, the transmission on broadcasting stations, video conferencing, wireless transmission etc..

Brief Description of the Drawings

Fig. 1 illustrates the computing environment of the preferred embodiment;

Fig. 2 illustrates the data flow over the network according to the preferred embodiment;

Fig. 3 is the flowchart of the method initially executed on the Media content server;

Fig. 4 is the flowchart of the method executed on the Application server;

Fig. 5 is the flowchart of the steps for preparing the dynamic HTML page, as part of the method executed on the Application server according to the preferred embodiment;

Fig. 6 is the flowchart of the method executed on the customer computer and the media content server, according to the preferred embodiment.

Detailed Description of the preferred embodiment

The present invention can be applied to any telecommunication network where there is one or more Media Content Server and one or more client that requests to use a media. In the following there is the detail of the preferred embodiment.

Fig.1 is a description of the environment of the invention. The customer computers (110, 120) are connected to the network (130) to access services provided by application servers (100). The application servers provide information to the customers who are their subscribers. In the case where the customer subscribes to a multimedia content data distribution service, the data itself is collected by the service provider from the media content owners. Usually, the media content data is stored on huge repository in media content servers (140) or accessed by the media content servers, accessible through the network.

The data flowing through the network and illustrated by dotted lines consist in messages exchanged between the subscriber and the application server to establish a session and messages exchanged between the subscriber and the various media content server to access the data itself. The application server is in charge to download the Browser media player which will allow the subscriber to play media content data on the computer.

In Fig.1, the application server is represented as physically separated from the various media content servers as in other embodiments the application server and media content server can be merged into one server.

In the preferred embodiment, the method of the invention is implemented as a servlet (150) in the application server and as a java program and java classes (180) in the Media

Content Servers. In the customer computer the method of the invention may be implemented as a media program plug-in associated to the browser (170). Receiving the client request, the Servlet (150) is activated on the Application server which opens the client session. Any other possible program operating on the data processing systems (110, 120, 100, 140) can implement the same method than the implementation (170, 160, 180) of the preferred embodiment. The communication protocol inside the networks has no impact on the method of the invention; HTTP can be used if the network is a TCP/IP network.

Fig. 2 illustrates the data flow between the customer computer (110) and the servers (100, 140). If the customer computer is connected to Internet, the browser sends an HTTP request to the server providing the media content distribution service. The HTTP request (200) includes a list of data the subscriber wants to access. For instance, the customer may require to get access to a list of video movies identified by clicking in a welcome web page displayed by the browser. The server is the 'Application server' as described in Fig. 1 but this same server may also contain the media content data bases and could act also as a media content server as stated also in reference to the same figure. The Application server performs the subscriber authentication, authorization which is not part of the preferred embodiment of the invention, and sends (210) to the customer browser web pages including the necessary information to retrieve the media content data. More particularly, the Application server creates a temporary metafile containing the information to retrieve the media content file address and the cyphered file paths corresponding to the media content data. The Application server sends (210) the metafile name to the customer computer. The metafile has a name depending on customer session parameters such as session number and creation date and time. In the preferred embodiment the name depends on the session number and the file creation

date and time to distinguish, with the date and time, between all the metafiles created during one same session. The life duration of the metafile name is the duration of the session just established. The customer computer has at least one media player plug-in added to the browser which will be used to send requests to the media content server and to be able to play the media content data files when received from the media content server. As a matter of fact the protocol between the customer computer and the media content server is specific, it is not the HTTP usual protocol when the network is an IP network. To each media content server corresponds a specific media player program.

The customer computer reads the metafile content (215) and sends a request (220) to the media content server to read and play the corresponding data included in the metafile. The read (215) and send (220) operations are operations usually performed by the media player program which is a plug-in, in the preferred embodiment, activated at reception of information (210) from the Application server. The media content server of the preferred embodiment has previously created a repository of media content data names which are cyphered file names according to a defined naming convention. Before answering the request of a customer according to the preferred embodiment, the media content server checks on the cyphered file name included in the request. If the file exists, the media content server downloads (230) the media content data which is played by the customer computer browser using the media player plug-in. The operation of read/request/dowload is repeated (215, 220, 230) for all the media content data files requested.

The request to stop the session is sent (240) by the customer computer to the Application server which suppresses temporary data related to this session, including the temporary metafile in its file repository.

Fig. 3 is the flow chart of the steps of the method initially executed in the media content servers according to the preferred embodiment. The execution is started (300) as a standalone task to create the environment which will allow the media content server to provide a secure distribution of the media content data files. A first step consists in calling a Java program (310), in the preferred embodiment, which will use a cyphering class (320). The execution of the cyphering class consists in cyphering the current media content data file names (330) as known from the system manager. Each file name follows a naming convention facilitating the identification of the file content by the file name. Conversely, a network cracker reading a video file name including the path, as listed by the browser of his computer, can retrieve one other file by using the naming convention. By cyphering the file name, the naming convention cannot be used by a cracker ignoring the cyphering process. The cyphering process used by the Java class according to the preferred is a standard one. The result of the cyphering step is the creation of a repository (340) in the media content server of media content data files having cyphered names.

Fig. 4 is the flow chart of the steps of the method executing on the Application server according to the preferred embodiment. This flow chart could describe the normal steps performed by the method executed on an application server giving access to subscribers for distribution of media content data from media content servers. However, as explained later in the document, the step (430) of creating dynamic HTML page is particular to the method of the preferred embodiment. An application is started on the Application server (400) able to handle the user log-in sessions (410). The user requests to use the media content data distribution services through an appropriate session of log-in (410). The Application server

collects user information and may performs authentication and authorization for the user who must be recognized as a subscriber. The Application server opens the session (420) storing customer related information and assigning, for instance, a session number. The servlet retrieves, from the customer request, the file names that the customer wants to access on the media content servers. The Application server has a table with the media content data file names and the file address in the network. For instance, the customer selects a title of movie on the browser capture screen and a name of file is automatically sent in the customer request. Using the table, the Application server has the address of these files. The next step (430) consists in preparing a dynamic HTML page including a variable metafile name. This will prevent the unauthorized users to access these information if they access the following exchange of messages the Application server will have with the customer. The details on the preparing step (430) are provided later in the document, in reference with Fig. 5. The following step (440) consists in sending this special dynamic HTML page to the customer.

Fig. 5 is the flow chart of the steps of the method to prepare the dynamic HTML page (430). An HTML page is a document written in an hypertext language that the browser installed on customer computers connected to the IP networks can interpret. HTML is used to transfer information and automatically starts execution of plug-in programs associated to the browser. In the Application servers of the prior art providing services for media content data distribution, a list of media content data file names is sent to the subscriber as well as the media content server address owning the files to allow the subscriber to connect to the media content servers and download the media content files. The HTML page built in step 430 will not include the file path names but will hide them in the following way:

- the customer has requested to log-in (410) the Application server opens the session (420) and receives the request from the subscriber of a list of media content data identified by a name. The application server, in the preferred embodiment reads a table providing a media content file address corresponding to a media content data name: for instance, the table contains a movie title and the file path of the corresponding video file; the file path includes the file name according to a naming convention and the path on the media content server owning that file;

- the Application server servlet calls a cyphering class which is known and used by the media content server having built the media content data file repository according to the method as described sooner in the document in reference to Fig. 3. The servlet of the Application server computes the cyphered file name (520) for each file of the list requested.

- the servlet calls a program to create a temporary metafile (530). A metafile, is a file containing information defining other files. For instance, the media player protocol of Real Networks Inc., uses '.ram' metafiles to store information on files to be played. According to the preferred embodiment, the name of the temporary metafile depends on temporary parameters such as the subscriber session parameters (such as a session number assigned by the Application server) and the date and time of session opening (420). The advantage of having a temporary file name is that a cracker who steals it included in a message can only use it during the duration of the session. As the file name is cyphered the origin of the file cannot be easily identified.

- the servlet of the Application server writes (540) the cyphered file path in the metafile and the address of the media content server containing these files;

- the servlet builds the HTML page including the metafile name (550) pointing to the media player plug-in that is used by the customer browser to receive the page request the media content

data files that the media content server will download and play it.

This is an example of HTML page built according to the preferred embodiment. The page contains video file references to be used by the customer computer browser and the media player plug-in for requesting and playing media content data files ("audio/x-pn-realaudio-plugin"). Instead of referring to explicit media content data file name and path, the page refers to a metafile name **(DW1EQ0LKL12BWM0ZBV25NBI.ram)** having a temporary file name, itself including a cyphered file name which does not reveal the naming convention on the media content server:

```
<EMBED SRC=/CMCache/DW1EQ0LKL12BWM0ZBV25NBI.ram"  WIDTH=160
HEIGHT=120  NOJAVA=true CONTROLS=ImageWindow CONSOLE=_master
type="audio/x-pn-realaudio-plugin"><br><br>
<EMBED SRC=/CMCache/DW1EQ0LKL12BWM0ZBV25NBI.ram"  WIDTH=170
HEIGHT=24 NOJAVA=true CONTROLS=StatusPanel CONSOLE=_master
type="audio/x-pn-realaudio-plugin"><br>
<EMBED SRC=/CMCache/DW1EQ0LKL12BWM0ZBV25NBI.ram"  WIDTH=16
HEIGHT=24 NOJAVA=true CONTROLS=RWCtrl CONSOLE=50k
type="audio/x-pn-realaudio-plugin">
<EMBED SRC=/CMCache/DW1EQ0LKL12BWM0ZBV25NBI.ram"  WIDTH=70
HEIGHT=24 NOJAVA=true CONTROLS=PositionSlider CONSOLE=50k
type="audio/x-pn-realaudio-plugin">
<EMBED SRC=/CMCache/DW1EQ0LKL12BWM0ZBV25NBI.ram"  WIDTH=16
HEIGHT=24 NOJAVA=true CONTROLS=FFCtrl CONSOLE=50k
type="audio/x-pn-realaudio-plugin">
<EMBED SRC=/CMCache/DW1EQ0LKL12BWM0ZBV25NBI.ram"  WIDTH=26
HEIGHT=24 NOJAVA=true CONTROLS=PlayButton CONSOLE=50k
type="audio/x-pn-realaudio-plugin">
<EMBED SRC=/CMCache/DW1EQ0LKL12BWM0ZBV25NBI.ram"  WIDTH=26
HEIGHT=24 NOJAVA=true CONTROLS=StopButton CONSOLE=50k
type="audio/x-pn-realaudio-plugin">
```

FIG. 6 the flow chart of the method of the preferred embodiment as executing on the customer computer receiving the HTML page built by the servlet as described sooner in the document in reference with Fig. 4 and Fig. 5.

Fig. 6 shows also the data flow between the customer computer, the media content server distributing the media content data files and the Application server.

The customer computer browser receives the HTML page. If the session corresponding to the metafile name is still maintained as active by the customer (answer Yes to test 610), the customer computer, activating a media player program already installed, reads the cyphered file path and media content server address in the metafile located on the application server, and sends (620), using a transfer protocol as expected by the media content server, the request to play the first media content data file. The media content server upon reception of the request, identifies the cyphered file name sent in the request to play from the customer. If the name is identified as correctly cyphered (answer yes to test 660), the file is taken from the data base and downloaded (670) according to the playing protocol through the network to the customer computer. The customer computer receives the data from the media content server and plays it using the media player plug-in. When the file playing is completed, if the customer maintains the session as active (answer yes to test 610), the next file stored in the metafile of the HTML page is requested to be played to the media content server owning it. To achieve this purpose, the next cyphered file name is sent from the customer computer to the media content server address which have been read in the metafile referred in the HTML page included in a new request.

If the media content server does not identify the cyphered name file (answer No to test 660) it sends an error

message. At any time, if the session is expired (answer No to test 610), the application server cancel any temporary data related to this session to close it including the temporary metafile (650). The metafile will be canceled as well from the file system repertory of the Application server.

Claims

1. A method for providing a subscriber computer with an access, through a network, to media content data from an application server, said method comprising the steps of:
- opening a session with a subscriber;
- receiving from the subscriber computer a request with a list of at least one media content data name;
- creating a temporary metafile having a temporary name;
- writing in the metafile a file path cyphered according to a cyphering algorithm for each media content data name and writing a corresponding network address where said media content data is stored;
- sending to the subscriber computer the temporary metafile name;
- canceling the metafile at the end of session with the subscriber computer.

2. The method of claim 2 further comprising the following initial step executed on the application server:
- creating a table, each entry containing a media content data name and its corresponding network address where the media content data is stored;
said method further comprising, before the writing step, the step of:
- reading in the table the corresponding network address for each media content data name.

3. The method of claim 1 or 2 wherein the creating step further comprise the step of:
- computing a variable metafile name as a function of subscriber session parameters and date and time of metafile creation.

4. The method of claim 3 further comprising an initial step of:

- defining a function, used in the computing step of subscriber session parameters and date and time of metafile creation.

5. The method of anyone of claims 1 to 4 further comprising the following steps executed on the subscriber computer:

- receiving the metafile name on the subscriber computer;
- reading the metafile on the application server;
- sending the request of playing each cyphered file path to the corresponding network address;
-receiving and playing the content of each file sent from the corresponding network address.

6. The method of anyone of claims 1 to 5 further comprising the following steps on a media content server having the corresponding network address:

- upon reception of the cyphered file path, checking, on said media content server that this cyphered file path is a file path of an existing media content file accessed by the media content server;
- if there is an existing media content file, sending this file for playing to the subscriber computer.

7. The method of claim 6 further comprising an initial step of:

- creating on the content media server a repository for media content data files accessed by said media content server, said repository comprising file path cyphered according to the same cyphering algorithm as the application server;

said method further comprising during the execution of the checking step, a step of:

- comparing the cyphered file path received from the subscriber computer with the cyphered file paths as named in the repository.

8. The method of anyone of claim 1 to 7 further comprising the following initial step executed on the subscriber computer:
- installing a media player program which is which automatically activated upon reception of the metafile name on the subscriber computer and which performs the steps of:
reading the metafile on the application server;
sending the request of playing each cyphered file path to the corresponding network address; and,
receiving and playing the content of each file sent from the corresponding network address.

9. The method of claim 8 wherein the step, performed on the application server, of sending to the subscriber computer the temporary metafile name, comprises the steps of:
- preparing an HTML page including an embedded command of the media player program pointing to the temporary metafile name;
- sending the HTML page to the subscriber computer;
and, wherein the step of receiving the metafile name on the subscriber computer, comprises the steps of:
- receiving the HTML page;
- activating the media player program on the metafile received in the HTML page.

10. A computer program product comprising programming code instructions for executing the steps of the method according to anyone of claims 1 to 9 when said program is executed on a computer.

11. A system comprising means adapted for carrying out the method according to anyone of claims 1 to 9.

A SECURE SYSTEM AND METHOD FOR MEDIA CONTENT DATA FILE NETWORK DISTRIBUTION

Abstract

A method is disclosed for providing a subscriber which wants to download and play a list of media content data from media content server in a secure way. The subscriber computer sends a request to an application server which will hide any path information allowing a cracker to retrieve media content data in the network. The application server writes cyphered media content data file paths and the network address of the media content server storing the media content data file paths in a temporary metafile having a name which depends on temporary parameters such as the subscriber session number and date and time of the metafile creation. The application server sends the metafile name to the subscriber computer.

The subscriber computer reads the cyphered media content data file paths on the application server in the metafile and sends the request to play them to the media content server. The media content server has used the same algorithm than the application server to cypher the data file path in its file repository; it retreives the files and send them for playing to the subscriber.

A cracker cannot retrieve the file path and learn the naming convention as they are cyphered in the metafile. The metafile, which is a temporary file, can no more be retrieved and used by a cracker after the termination of the subscriber session with the application server.

Fig. 1

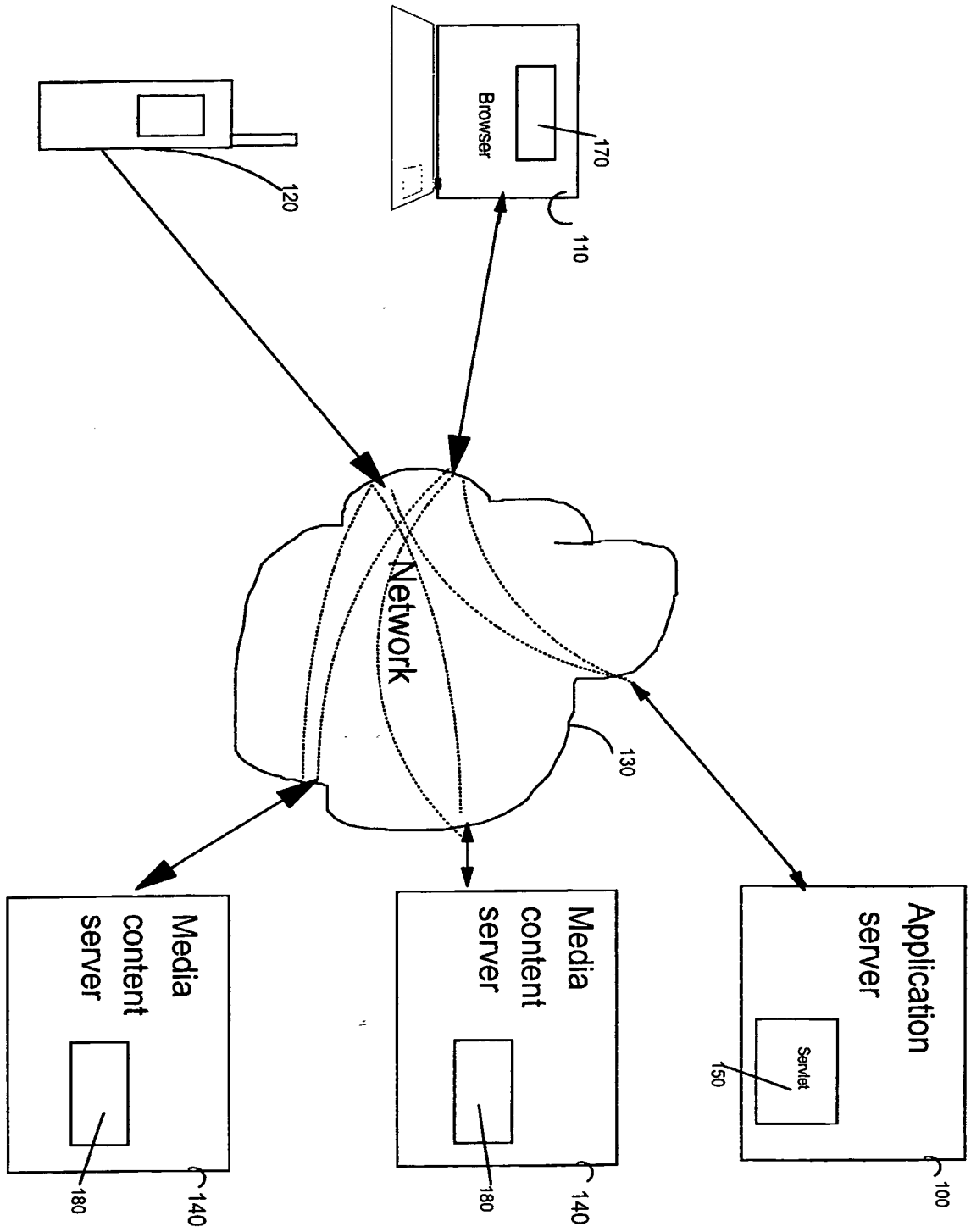
Browser
170
110
120
Network
130
Application server
Servlet
150
100
Media content server
180
140
Media content server
180
140

FIGURE 1

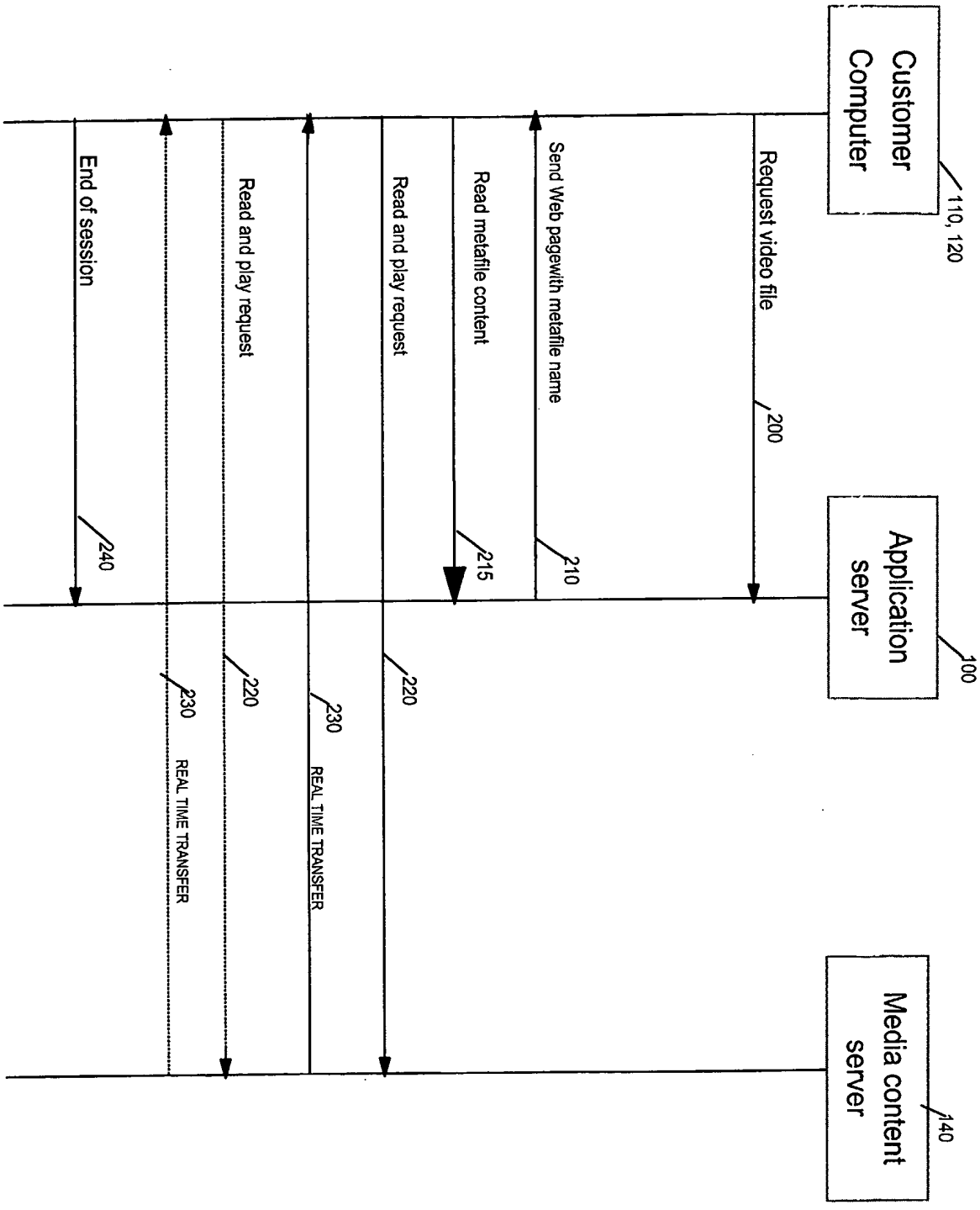
Customer Computer
110, 120
Application server
100
Media content server
140
Request video file
200
Send Web pagewith metafile name
210
Read metafile content
215
Read and play request
220
REAL TIME TRANSFER
230
Read and play request
220
REAL TIME TRANSFER
230
End of session
240

FIGURE 2

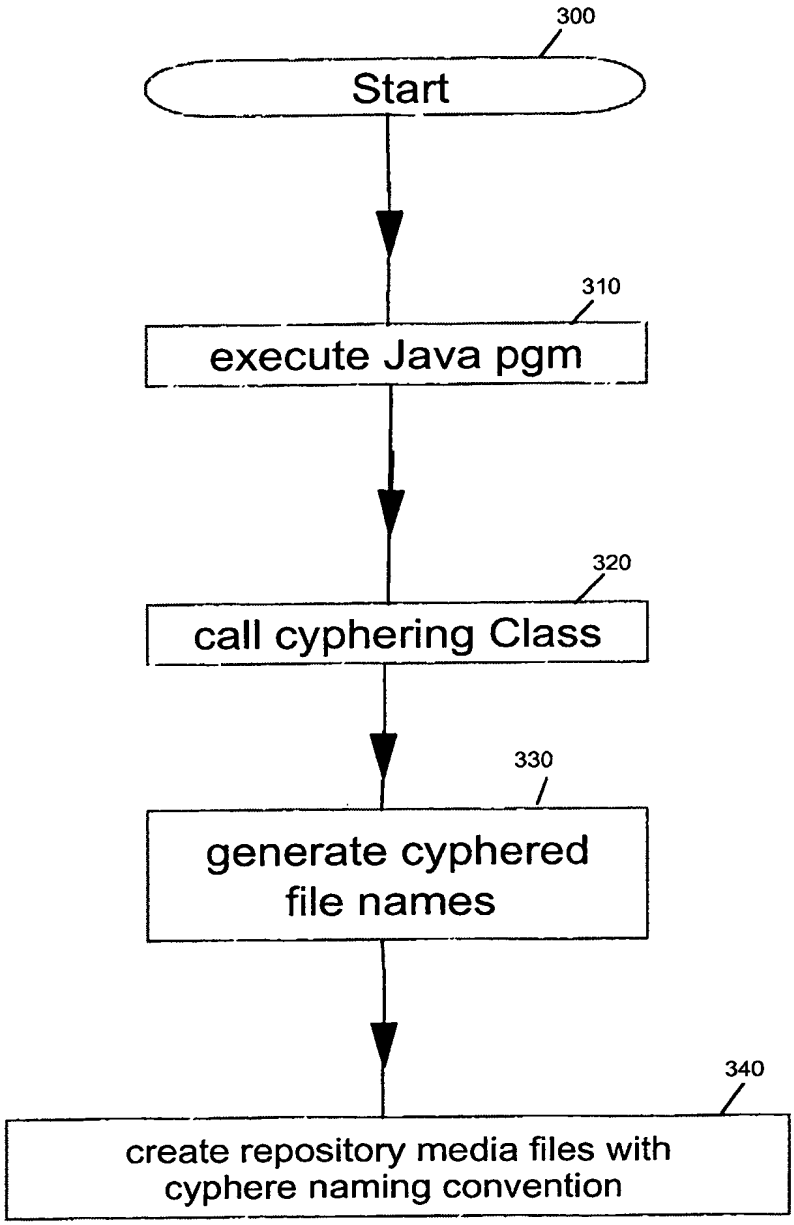
300
Start
310
execute Java pgm
320
call cyphering Class
330
generate cyphered file names
340
create repository media files with cyphere naming convention

FIGURE 3

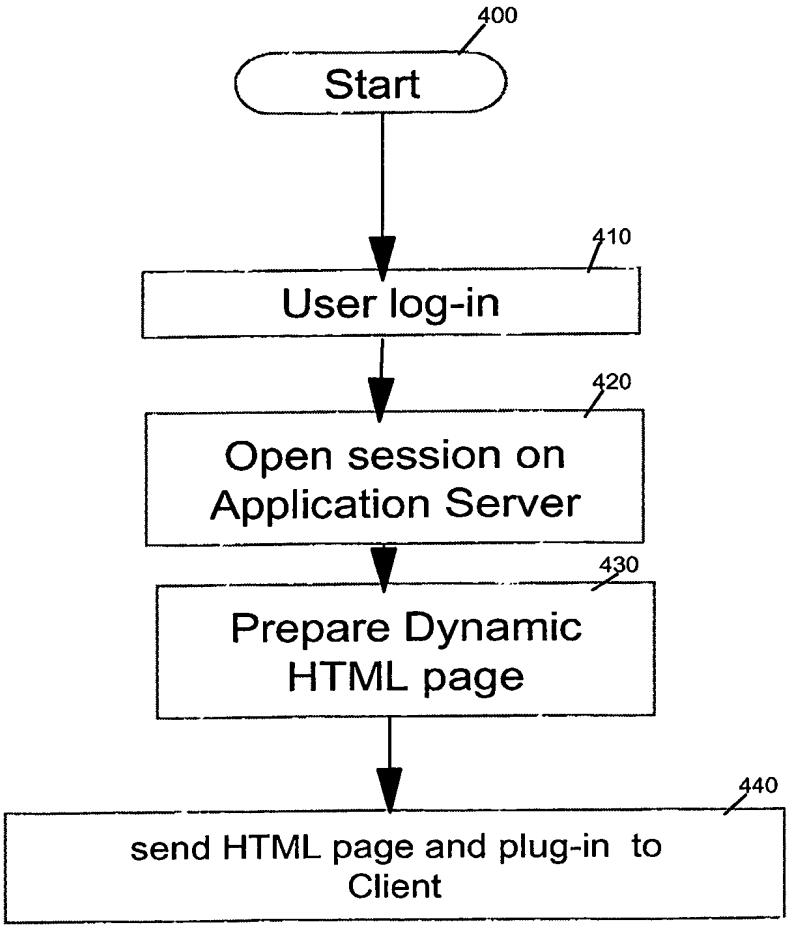
400
Start
410
User log-in
420
Open session on Application Server
430
Prepare Dynamic HTML page
440
send HTML page and plug-in to Client

FIGURE 4

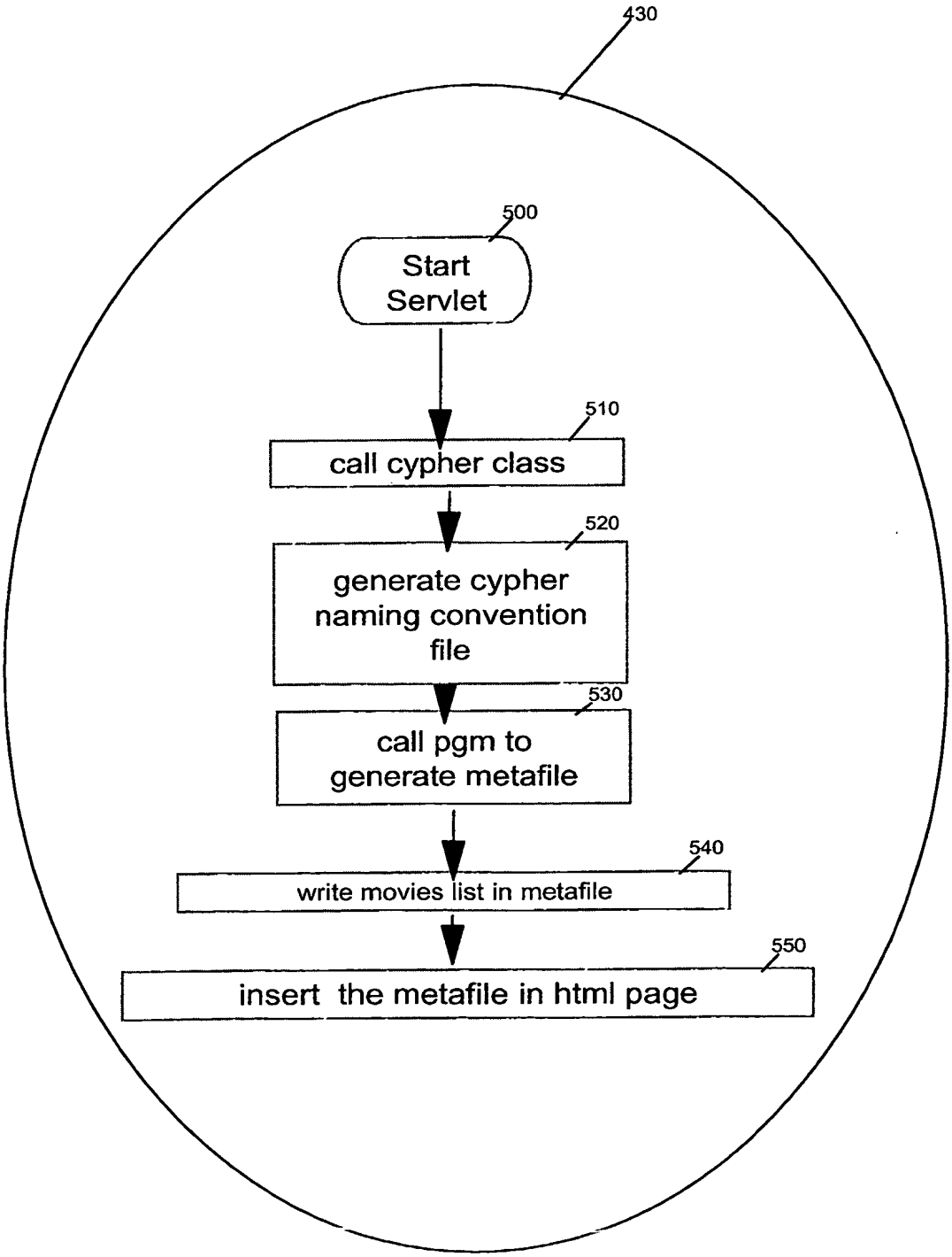
430
500
Start Servlet
510
call cypher class
520
generate cypher naming convention file
530
call pgm to generate metafile
540
write movies list in metafile
550
insert the metafile in html page

FIGURE 5

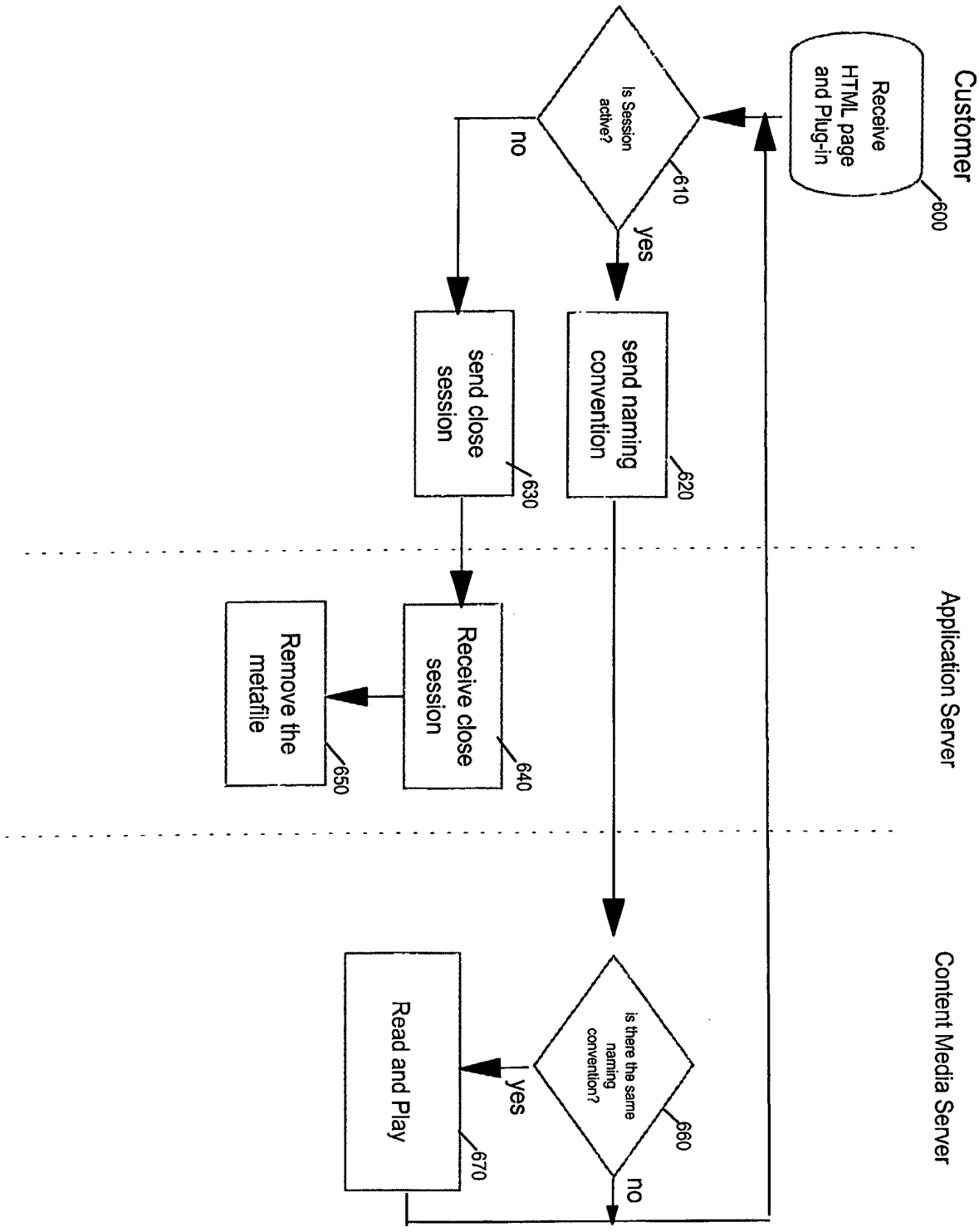
Customer
Application Server
Content Media Server
Receive HTML page and Plug-in
600
Is Session active?
610
no
yes
send naming convention
620
send close session
630
Receive close session
640
Remove the metafile
650
is there the same naming convention?
660
yes
no
Read and Play
670

FIGURE 6